Advanced Topics in Compilers

Induction variables

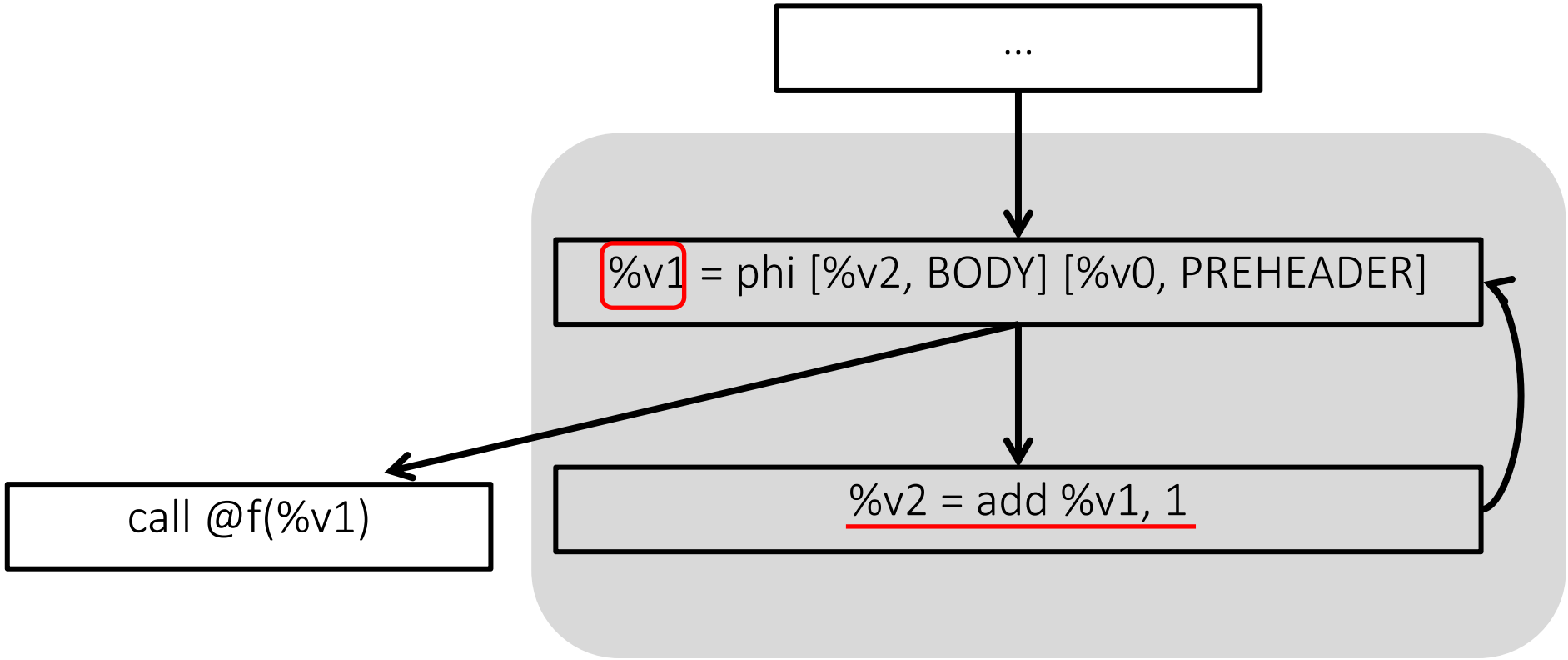
Simone Campanoni
simone.campanoni@northwestern.edu

Outline

- What is it and why NOELLE provides it
- Checking induction variables
- Loop governing induction variable

InductionVariablesManager

- It captures the induction variables of a loop



Outline

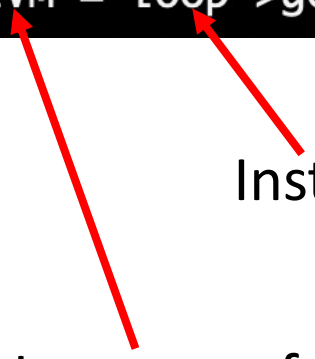
- What is it and why NOELLE provides it

- Checking induction variables

- Loop governing induction variable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```



Instance of the class arcana::noelle::LoopContent

Instance of the class arcana::noelle::InductionVariablesManager

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
 }
```

```
/*
 * Print the main PHI of the IV
 */
errs() << "   IV: " << *IV->getLoopEntryPHI() << "\n";
```

Instance of the class arcana::noelle::InductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
}
```

```
/*
 * Print the PHIs
 */
errs() << "     PHIs that compose the SCC of the IV\n";
auto phis = IV->getPHIs();
for (auto phi : phis){
  errs() << "       " << *phi << "\n";
}
```

Instance of the class arcana::noelle::InductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
 }
```

```
/*
 * Print the type of the variable that behaves like an IV
 */
auto t = IV->getType();
errs() << "     Type of the IV: " << *t << "\n";
```

Instance of the class arcana::noelle::InductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
 }
```

```
/*
 * Print the start value of the IV
 */
auto s = IV->getStartValue();
errs() << "     Start value = " << *s << "\n";
```

Instance of the class arcana::noelle::InductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
}
```

```
/*
 * Print the sequence of computation steps that computes the delta that is applied to each update of the IV.
 */
errs() << "     Sequence of computation steps that computes the delta that is applied to each update of the IV:\n";
for (auto stepValueComputationInstruction : IV->getComputationOfStepValue()){
  errs() << "       " << *stepValueComputationInstruction << "\n";
}
```

Instance of the class arcana::noelle::InductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Iterate over all induction variables.
 */
errs() << " Induction variables\n";
for (auto IV : IVM->getInductionVariables()) {

39 lines: Print the main PHI of the IV----------
}
```

```
/*
 * Print the SCC
 */
auto scc = IV->getSCC();
errs() << "     SCC has " << scc->numberOfInstructions() << " number of instructions\n";
```

Instance of the class arcana::noelle::InductionVariable

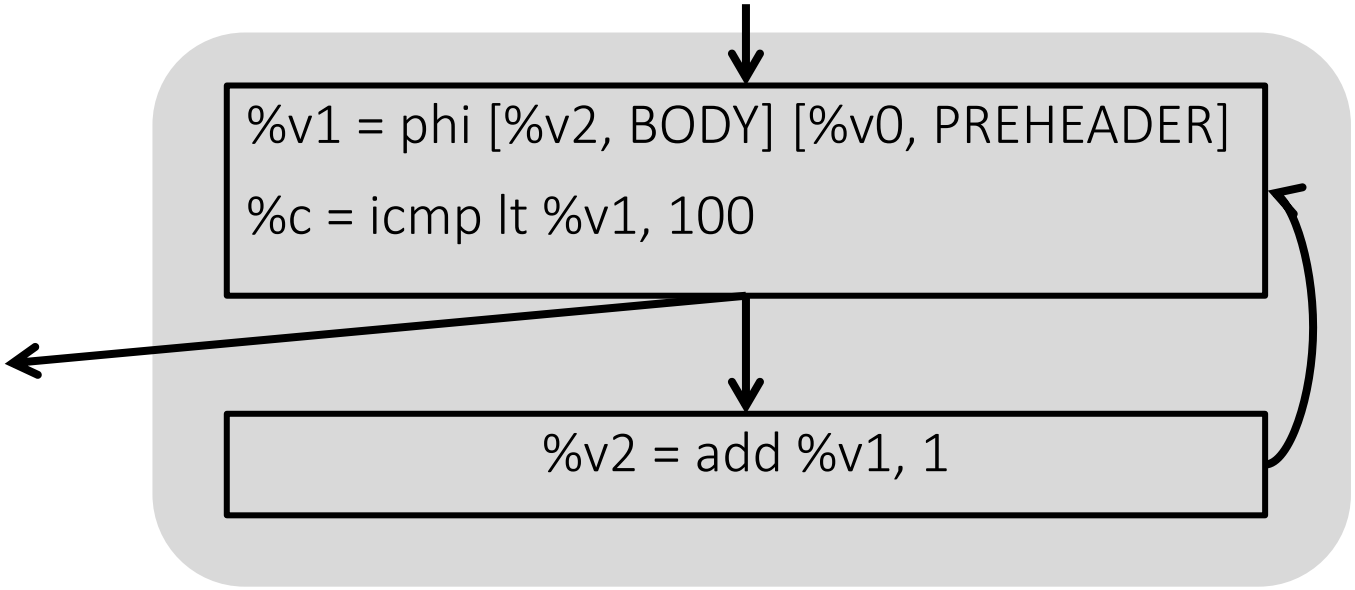
Outline

- What is it and why NOELLE provides it
- Checking induction variables
- Loop governing induction variable

Loop governing induction variable

It is an induction variable that

- controls how many iterations will execute
- nothing else will control when to leave the loop

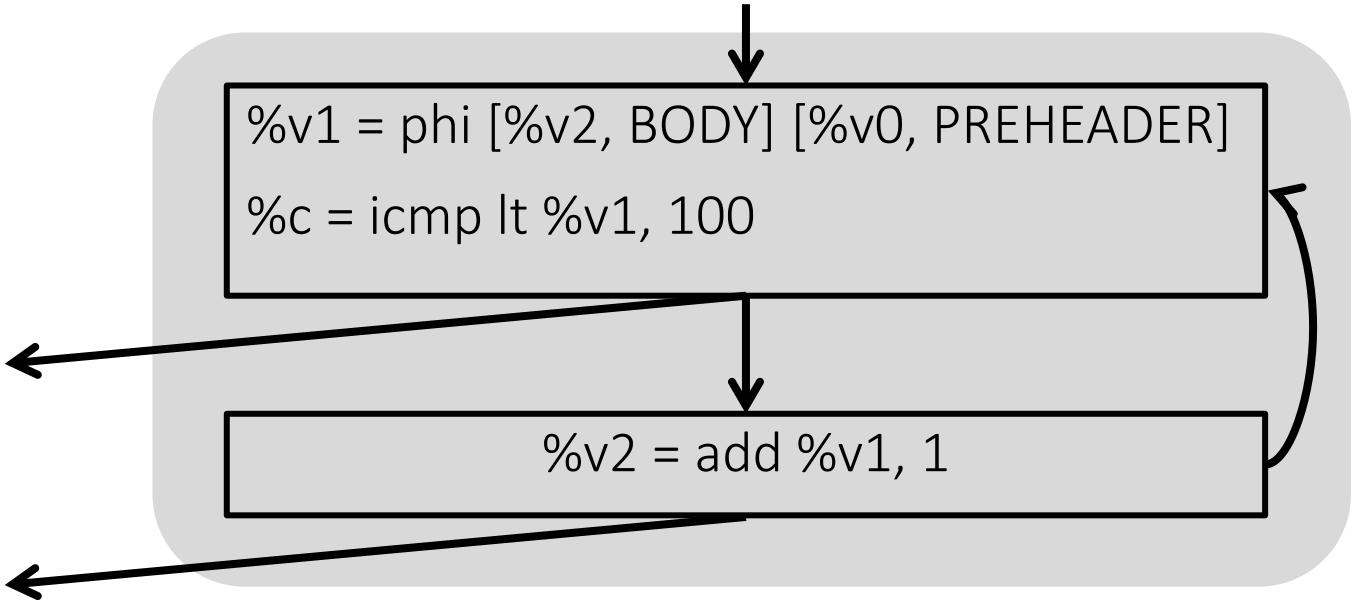


%v1 is a loop governing IV

Loop governing induction variable

It is an induction variable that

- controls how many iterations will execute
- nothing else will control when to leave the loop



%v1 is not a loop governing IV

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Print the loop governing induction variable
 */
auto GIV = IVM->getLoopGoverningInductionVariable();
if (GIV != nullptr) {
  errs() << "   The loop has a loop governing IV\n";
}
```

Instance of the class arcana::noelle::LoopGoverningInductionVariable

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Print the loop governing induction variable
 */
auto GIV = IVM->getLoopGoverningInductionVariable();
if (GIV != nullptr) {
  errs() << "   The loop has a loop governing IV\n";
}
```

Instance of the class arcana::noelle::LoopGoverningInductionVariable

```
/*
 * Print the exit condition
 */
auto exitCondition = GIV->getExitConditionValue();
errs() << "    Exit condition = " << *exitCondition << "\n";
```

Instance of the class llvm::Value

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Print the loop governing induction variable
 */
auto GIV = IVM->getLoopGoverningInductionVariable();
if (GIV != nullptr) {
  errs() << "   The loop has a loop governing IV\n";
}
```

Instance of the class arcana::noelle::LoopGoverningInductionVariable

```
/*
 * Print the value to compare against the exit condition.
 */
auto valueToCompare = GIV->getValueToCompareAgainstExitConditionValue();
errs() << "    Evolving value to compare = " << *valueToCompare << "\n";
```

Instance of the class llvm::Instruction

Checking induction variables

```
/*
 * Induction variables.
 */
errs() << " Induction variables\n";
auto IVM = loop->getInductionVariableManager();
```

```
/*
 * Print the loop governing induction variable
 */
auto GIV = IVM->getLoopGoverningInductionVariable();
if (GIV != nullptr) {
  errs() << "   The loop has a loop governing IV\n";
}
```

Instance of the class arcana::noelle::LoopGoverningInductionVariable

```
auto IV = GIV->getInductionVariable();
```

Instance of the class arcana::noelle::InductionVariable

Always have faith in your ability

Success will come your way eventually

Best of luck!